VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Detekce objektů v obrazech s využitím markerů na platformě Android

Marker-based Object Detection on Android

2018 Petr Noga

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Petr Noga**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Detekce objektů v obrazech s využitím markerů na platformě Android
Marker-based Object Detection on Android

Jazyk vypracování: čeština

Zásady pro vypracování:

Detekce a rozpoznání objektů může být velmi náročný úkol, zejména pokud se jedná o detekci objektů, které jsou různě deformované nebo různě natočené. K nalezení a rozpoznání objektů může v takovém případě pomoci detekce založená na obrazových markerech.

1. Seznamte se se základními pojmy v oblasti detekce markerů za pomocí obrazů.
2. Seznamte se s knihovnami z oblasti zpracování obrazů (například OpenCV).
3. Implementujte detektor markerů na platformě Android (využít můžete nastudované knihovny).
4. Experimentálně ověřte funkčnost, přesnost a rychlost detektoru.
5. Své závěry řádně zdokumentujte v textu práce.

Seznam doporučené odborné literatury:

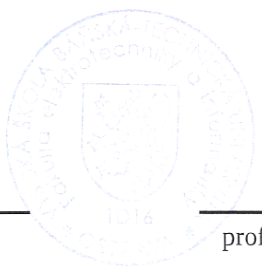
[1] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. 2014. "Automatic generation and detection of highly reliable fiducial markers under occlusion". Pattern Recogn. 47, 6 (June 2014), 2280-2292

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radovan Fusek, Ph.D.**

Datum zadání: 01.09.2017

Datum odevzdání: 30.04.2018



doc. Ing. Jan Platoš, Ph.D.
vedoucí katedry

prof. Ing. Pavel Brandštetter, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 30. dubna 2018 .....

Rád bych na tomto místě poděkoval všem, kteří mě podporovali v psaní bakalářské práce. Hlavně bych chtěl poděkovat panu Ing. Radovanovi Fuskovi, Ph.D. za příkladné vedení a cenné rady v průběhu tvorby této bakalářské práce.

Abstrakt

Cílem této bakalřáské práce je seznámení se se základními pojmy v oblasti detekce aruco markerů. V bakalářské práci se zabývám vysvětlením jednotlivých pojmů v oblasti rozšířené reality a popisem jednotlivých kroků pro realizaci detekce markerů. Před samostatnou detekci markerů, se musí provést převod snímku černobílý snímek. Po převodu se provádí binární prahování a detekce obrysů z markeru. Po správné detekci obrysů se musejí nalézt z obrazu kandidáti na marker. Před samostatným umístění markeru v prostoru se musí ověřit zda je to marker, nebo né a to extrahováním bitů z markeru. Na konci této práce popisuji aplikaci, která může být využita ve spoustě lidských činností od údržby automobilu až po výukové účely.

Klíčová slova: OpenCV, markery, pozice markeru,kalibrace kamery, aruco, detekce markerů

Abstract

The aim of this bachelor thesis is to get acquainted with the basic concepts in aruco marker detection. The bachelor thesis deals with the explanation of individual concepts in the field of extended reality and description of the individual steps for realization of detection marker. Before seperate detection of markers, a black-and-white image conversion must be performed. After conversion, binary thresholding and marker contour detection are performed. Once the contours have been correctly detected, the marker candidate must be found in the image. Before separating the marker in the space, it must be checked whether it is a marker or not by extracting bits from the marker. At the end of this work I describe an application that can be used in a lot of human activities from car maintenance to teaching purposes.

Key Words: OpenCv, markers, Pose Estimation, camera calibration, aruco, marker detection

Obsah

Seznam použitých zkratek a symbolů

AR	–	Rozšířená Realita
SDK	–	Software Development Kit
HMD	–	head-mounted display
SLAM	–	Simultaneous Localization And Mapping
CPP	–	Programovací jazyk C++
AVA	–	Applications of Artificial Vision
BSD	–	Berkeley Software Distribution
GPS	–	Global Positioning System
STL	–	Standard Template Library

Seznam obrázků

Seznam tabulek

Seznam výpisů zdrojového kódu

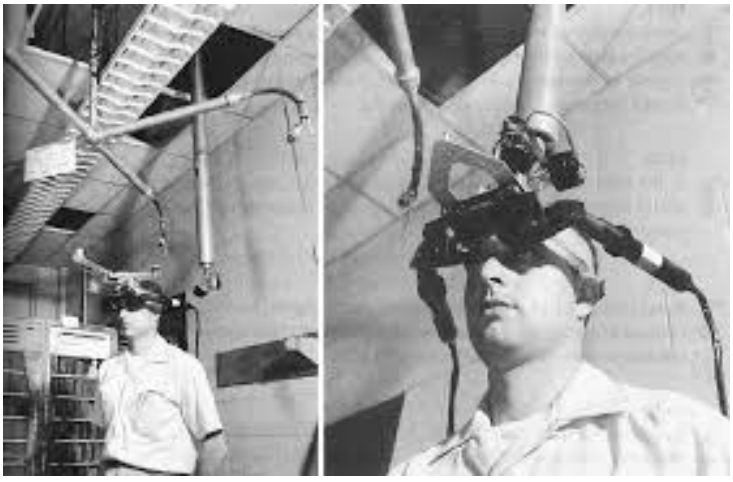
1 Úvod

Aplikace pro detekci objektů v obrazech se v dnešní době používají na spousty místech. Čím dál tím více se detekce dostává do podvědomí lidí i mimo vědeckou sféru a pro jejich využití se značně rozšiřují. Virtuální realita nepoužívá reálné prostředí, ale podle Amerického počítačového odborníka Jarona Laniera "počítačem vytvořené interaktivní trojrozměrné prostřední, do něhož se člověk totálně ponoří"[1]. Rozšířená realita je technologie, která v reálném čase rozšiřuje dosavadní realitu o prvky digitální jako jsou např. texty, 2D objekty nebo 3D objekty. Mezi digitální prvky rozšířené reality patří například nábytek, který bude zakoupen z katalogu nábytků. Jelikož není známo, jak by tento nábytek vypadal v místnosti použije se aplikace, která používá pro své účely fotoaparát a namířením na místo, kde bude položen marker se zobrazí vybraný nábytek.

Marker je značka, která obsahuje informace pro aplikace. Markerům bude věnována samostatná kapitola 1.4. Existuje mnoho frameworků, které napomáhají rozšířit realitu jako jsou Wikitude [2], Vuforia[4] a OpenCV[3]. Každý z těcho frameworků detekuje své vlastní markery. Pro svou bakalářkou práci jsem použil Aruco framework z knihovny OpenCV, jelikož je velmi snadný na implementaci.

1.1 Historie AR

V roce 1965 Americký vědec Ivan Sutherland vynalezl head-mounted displej, který nazval jako The Sword of Damocles , který je zobrazen na obrázku 1. Pomocí tohoto zařízení v reálném světe pozoroval jednoduchou virtuální krychli, čímž vytvořil první rozhraní AR (rozšířené reality). U rozšířené reality není potřeba, aby programátor vytvářel nový svět, ale pouze vzal reálné prostředí, a to obohatil o prvky virtuální (tzn. umělé dokomponované objekty).



Obrázek 1: The Sword of Damocles [5]

Podle Ronalda T. Azumy se rozšířená realita definuje podle tří vlastností:

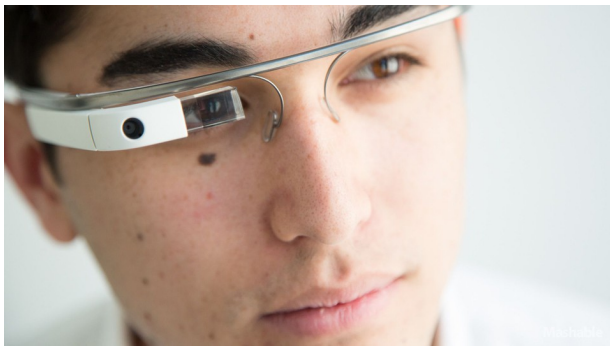
- kombinuje běžnou realitu a virtuální realitu,
- je interaktivní v reálném čase,
- využívá reálné 3D prostředí.[6]

1.2 Technické vybavení

K tomu, aby bylo možné vykreslovat rozšířené prvky v realitě potřebujeme zobrazovače, ty lze rozdělit na dva typy:

- head-mounted display,
- handheld display.

Head-mounted display je zobrazovací zařízení připevněno k hlavě nebo je součástí helmy a obsahuje malý displej před jedním nebo oběma očima. Rozděluje se na 2 typy podle toho zda je přes displej vidět nebo není. Pokud lze přes displej vidět, lze tento HMD rozlišit na opticky průhledový displej a vykreslování obrazů na sítnici, to je možné vidět na obrázku 2a. V druhém případě je možné vidět na obrázku 2b zařízení od Sony s displejem. Na displeji se promítá video.



(a) Google Glass [7]

(b) Sony Head mounted display [8]

Obrázek 2: Head-mounted Displays

Handheld display (kapesní displej) je používán v malých výpočetních zařízeních. Mohou to být telefony, tablety nebo také PDA. Handheld displays používají technologi Video see-through, mobilem generovaný obraz se mísí s obrazem reálného světa z mobilní kamery. Následně je tento obraz prezentován na mobilním displeji. Dále se využívají senzory pro zjištování polohy uživatele, tzn. digitální kompasy. V dnešní době, kdy jdou technologie velmi rapidně dopředu máme na trhu nespočet mobilních telefonů, které mají velmi dobře zpracované a výkonné komponenty v tomto případě jde např. o procesor, paměti, kameru apod. Pomocí těchto komponent vývojáři mohou vytvářet propracovanější AR aplikace. Na obrázku 3 je zobrazen mobilní telefon, na kterém jsou u objektů promítany prvky rozšířené reality v podobě textu a obrázků.

Obrázek 3: Mobilní telefon s rozšířenou realitou [9]

1.3 Oblast využití AR

Rozšířená realita se ubírá oblasti vizuální. Virtuální realita klade důraz na hmat i sluch. AR v dnešní době je velmi populární nejen v herním průmyslu, ale také v medicíně, vojenském průmyslu, plánování, letectví, navigace.

Z technologického hlediska lze virtuální realita rozdělit do dvou oblastí:

- rozšiřování reality o virtuální informace,
- programování markerů.

V oblasti průmyslu se doplnění reality o virtuální informace využívá pro lepší orientaci svářečů a montérů, tuto technologii jako jedna z prvních firem využila automobilka BMW [10]. V tomto případě se využívá optika jako head-mount display, popřípadě přidavné senozory pro získávání dalších informací, jako je váha objektů nacházejících se v okolí dělníka.

V herním průmyslu v roce 2016 společnost Niantic vytvořila hru jménem Pokemon GO, která je založená na principu rozšířené reality. Prostřednictvím aplikace propojuje herní prostředí s reálným světem v reálném čase - na mobilní obrazovce se hráči ukáže mapa okolí, kde se nachází. Aplikace používá hráčskou polohu k objevování pokémonů v rozšířeném světě, a to na mapě v okolí hráče. Pokud se dostane do boje s jedním z těchto pokémonů je jeho úkolem pomocí

pokebalu chytit pokémona. Hráč si může vybírat v jakém prostředí bude bojovat, buď bude bojovat ve virtuálním prostředí nebo v rozšířeném prostředí. V případě rozšířeného prostředí se zapne kamera a na obrazovce mobilního telefonu hráče se zobrazí reálné prostředí s rozšířeným prostředím (pokemonem) jak je možné vidět na obrázku 4.

Obrázek 4: Zobrazení pokémona v rozšířené realitě [11]

V oblasti automobilového průmyslu se v posledních letech začínají využívat HUD technologie. Jak je zobrazeno na obrázku 5. Tyto technologie dokáží na čelním skle u řidiče zobrazovat informace například o jeho rychlosti, informace o projíždějících místech (např. benzínové stanice, restaurace a obchodní domy).[12] Tato technologie napomáhá řidičům, aby se věnovali hlavně řízení a dívali se dopředu na cestu. HUD displej se umístí na čelní sklo hned před řidiče blíže k palubní desce. Toto opatření má zamezit rozptýlení řidiče jako je tomu u používání obyčejné navigace, kterou má řidič umístěnou uprostřed palubní desky a musí se otáčet, aby viděl, kde má odbočit či jet.

Obrázek 5: Hud Display [13]

1.4 Markery

Marker je obrázek s jedinečným vzorem, který může být zachycen kamerou a rozpoznán softwarem. Pro rozšířenou a virtuální realitu bývá největším problémem odhad kamery v prostoru. Pro možnost umístění objektu do prostoru je potřebné znát polohu a orientaci markeru v prostoru.

Pro získání těchto informací z kamery je potřeba nalézt korespondenci mezi body v reálném prostředí a body kamery. Markery jsou atraktivním přístupem, protože jsou snadno rozpoznatelné a umožňují dosáhnout vysoké rychlosti a přesnosti detekce.

Markery se dělí na:

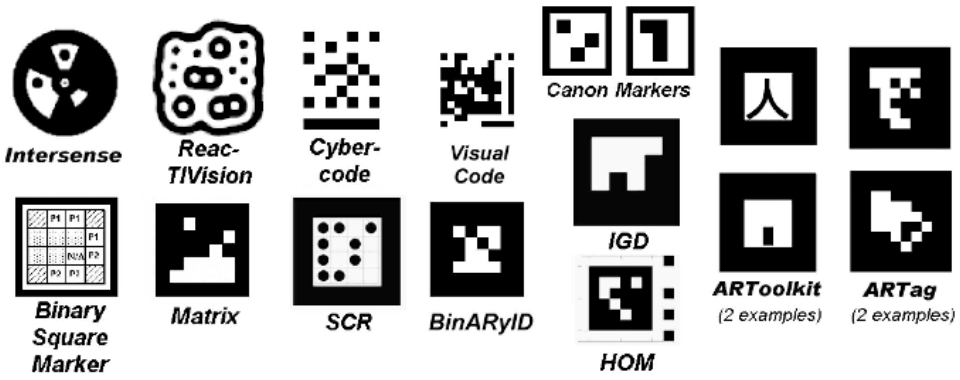
- markery,
- obrázkové markery,
- GPS markery,
- objektové markery,
- sledování,
- multimarkery.

Marker je obrazec, který je vytištěn na papír nebo jiný druh materiálu. Tyto markery jsou běžně čtvercovitého tvaru s černým (někdy bílým) ohraničeným okrajem. Během sledování software vyhledává černé čtverce, v tomto případě stačí pokud nalezne jen jeden, poté software zkoumá vnitřek rámečku pro určení skutečného markeru. Software extrahuje polohu a rotaci markeru v závisloti na tom jak je ohraničený okraj ve vztahu k fotoaparátu.

Obrázkové markery mají výhodu v tom, že namísto rámečkových markerů si uživatel může vytvořit svůj vlastní marker. Při detekci vlastního markeru dochází k větším chybám než při detekci obyčejného rámečkového markeru.

Sledování je technika, kdy se nemusí používat žádný marker. Tuto techniku používají Kudan[14] a ARCore[15] frameworky. Uživatel nejprve pomocí aplikace umístí manuálně objekt na libovolné místo v prostoru. Může se zdát, že díky senzorům systém nechává objekt uzamčen na místě. Je pravdou, že v dnešní době systémy pokročily vpřed v přesnosti, ale stále to nestačí. Informace senzoru jsou kombinovány s řešením problému SLAM (Simultaneous Localization And Mapping). Když uživatel umístí objekt "hololeny" vybírají funkční body kolem objektu a ze zadu objektu. Pro nastavení polohy objektu použijí tyto informace ve spojení se senzorovými informacemi.

Na obrázku 6 jsou zobrazeny markery společností, které vyvíjí své SDK pro rozšířenou realitu. Každá společnost se chtěla odlišit od ostatních společností. Proto firmy vytvářely své vlastní markery.



Obrázek 6: Každá společnost má vůj vlastní marker. [16]

2 Srovnání frameworků

Na trhu existuje mnoho frameworků pro práci s rozšířenou realitou. Jedním z nejpoužívanějších a zatím nejlepších frameworků je Vuforia. Oproti němu stojí sada knihoven OpenCV, která nepatří mezi nejpoužívanější knihovny, ale poskytuje robustní řešení. OpenCV má samostatnou kapitolu 3.1.

2.1 Vuforia

Vuforia je jedna z nejpoužívanějších knihoven na trhu a je volně šiřitelná a má tyto funkce:

- rozpoznává odlišné objekty jako např. krabice, letadla, válce,
- rozpoznávání textu,
- rozpoznávání prostředí,
- podporuje detekci VuMark markerů obrázek 7,
- podpora pro skenování objektů pomoci aplikace.

Obrázek 7: VuMark marker společnosti Vuforia. [17]

2.2 Wikitude

Wikitude není volně dostupná knihovna, programátor si ji musí zakoupit podle ceníku na stránkách wikitude. Framework se hlavně zaměřuje na lokalizační AR. Má tyto funkce:

- 3D sledování,
- rozpoznávání objektů,
- rozpoznávání obrázků,
- podporuje i jiné frameworky jako ARCore, ARKit.

3 Aruco knihovna a OpenCv knihovna

3.1 OpenCV

OpenCV je Open Source knihovna počitačové vize a strojového učení. Poskytuje společnou infrastruktůru pro aplikace počítačového vidění a urychluje používání vnímání stroje v komerčních produktech. Společnost OpenCV, která je licencovaným produktem BSD usnadňuje podnikům využívat a upravovat kód dle jejich potřeby. Tato knihovna má více než 2500 optimalizovaných zdrojových algoritmů, které zahrnují komplexní sadu klasických a nejmodernějších počitačových vizí a algoritmů pro strojové učení. Algorimty lze použít k detekci a rozpoznávání obličejů, identifikace obličejů, klasických lidských akcí ve videu, sledování pohybu kamer, sledování pohyblivých objektů, extrahování 3D modelů objektů, rozpoznávání markeru pro rozšířenou realitu.[18]

Má rozhraní C++, Python, Java a Matlbab. Podporuje Windows, Linux, Android a Mac Os. OpenCV knihovna se většinou opírá o aplikace v reálném čase. OpenCV je napsána nativně v C++ a má šablonové rozhraní, které pracuje hladce s STL kontejnery.

3.2 Aruco

Výzkumná skupina "Applications of Artificial Vision"(AVA) byla založena v roce 1998 třemi profesory z univerzity Cordoba. Tito profesoři se především zajímají o oblast umělého vidění a jeho aplikace. Tato skupina vyvíjela spousty projektů jako například 3D skenování objektů, automatické rozpoznání reakce člověka, ale hlavně se zabývají rozšířenou realitou, proto vyvinuli svou vlastní knihovnu pod názvem Aruco. Tato knihovna je licencováná jako Open source (volně přístupná). Výhodou je oproti jiným společnostem, které vyvíjí své vlastní SDK nebo Knihovny pro rozšířenou realitu to, že si ji programátor může upravit podle sebe a je zdarma.[19]

Pro svou bakalářkou práci jsem si vybral tuto knihovnu, protože je volně přístupná. Knihovna je naprogramována v C++ a může se použít jako nativní knihovna, tzn. tuto knihovnu je možné použít nativně s Javou, takže část kódu bude naprogramován v Javě a zbytek v C++. Pro detekci markerů je zapotřebí použít tyto knihovny z OpenCV knihovny:

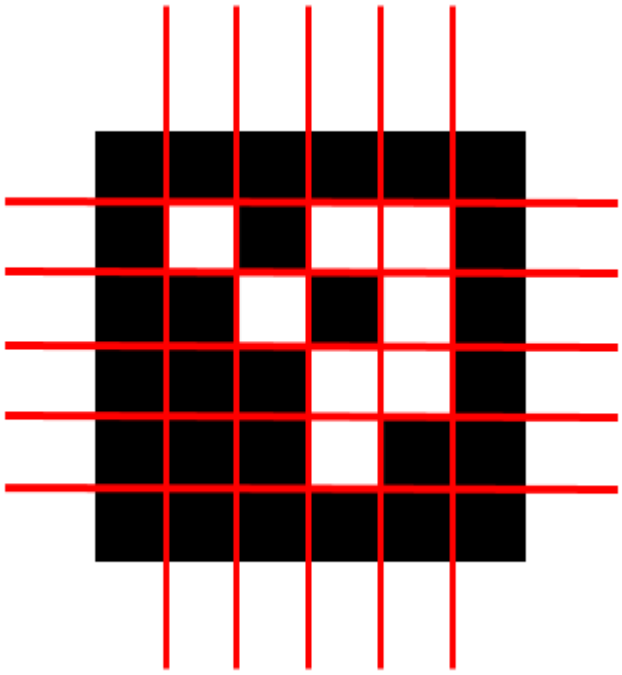
- opencv2/opencv.hpp
- opencv2/aruco.hpp
- opencv2/highgui.hpp

3.3 Aruco Marker

Motivací výzkumné skupiny AVA, bylo vymyslet takový marker, který bude jednoduchý a snadno detekovaný. Aruco knihovna podporuje až 1024 markerů, které si programátor může sám vygenerovat. Marker je čtvercový obrazec, který se skládá z mřížky a rohů pro správnou orientaci v prostoru. Mřížka je rozdělená na sloupce a řádky. Bílé políčko je bit s hodnotou 1 a černé políčko je bit s hodnotou 0.

Tuto mřížku je možné měnit pomocí předefinovaného slovníku. Slovník definuje parametry markeru.

Na obrázku 8 je marker ze slovníku 6x6_50 kde 6x6 je rozdělení markeru na 6 řádků a 6 sloupců a číslo _50 udává velikost slovníku tzn. kolik markerů je obsaženo ve slovníku 6x6. Pro aplikaci, která bude detekovat například 10 markerů je lepší použít slovník s menším číslem než větším. Je dokázané, že čím větší číslo slovníku tím je větší chybovost při prováděné detekci markerů. Marker má své vlastní identifikační číslo a vlastní jedinečný obrazec. Výhodou markerů je jejich velikost a tvar, jsou jednodušší a na detekci snadnější. Protože marker je snadněji detekovaný než složitý objekt, umístí se na místo složitého objektu jako je motor, chladič atd. jednoduchý dobře a rychle detekovatelný marker.



Obrázek 8: Marker 6x6 [20]

3.4 Vygenerování markeru

Vygenerování markeru probíhá před tím než se začne detekovat, proto je zapotřebí vytisknout marker na papír a umístit do reálného prostředí např. na stůl. Obrázky markeru lze generovat pomocí funkce drawMarker(). Ve výpise 1 je možné vidět část kódu pro vygenerování markeru s ID 0.

```
cv::Mat markerImage;
cv::Ptr<cv::aruco::Dictionary> dictionary = cv::aruco::
    getPredefinedDictionary(cv::aruco::DICT_6x6_50);
cv::aruco::drawMarker(dictionary,0,200,markerImage,1);
```

Výpis 1: Příkazy pro vygenerování markeru

Zde popíšu část kódu. Vytvořím si proměnou markerImage typu Mat, který je standardní typ OpenCV pro prací s obrazem. Tato proměnná bude výstupní obrázek (vygenerovaný marker). Mat je matice bodů, resp. jednotlivých barevných složek obrazu. Slovník je vytvořen vybráním jedné z předefinovaných slovníků v aruco modulu, k tomuto slouží funkce getPredefinedDictionary() s parametrem cv::aruco::DICT_6x6_50. Konkrétně tento slovník obsahuje 250 markerů a jeho velikost je 6x6 bitů (DICT_6x6_50). Poté pomocí funkce drawMarker() vygeneruji a uložím marker na disk, parametry pro funkci popíši zde:

- prvním paramterem je nadefinovaný slovník,
- druhým parametrem je identifikační číslo markeru, generuje se marker s ID 0 ze slovníku DICT_6x6_50. Každý slovník má své identifikační čísla a mohou se generovat markery s ID od 0 do 49. Pokud bude číslo vyšší než 49 aplikace vypíše výjimku.
- třetí parametr určuje velikost výstupního obrázku v pixelech, v tomto případě bude mít výsledný obrázek velikost 200x200 pixelů. Tento parametr by měl být dostatečně velký pro uložení počtu bitů pro daný slovník. Například nelze vytvořit obrázek o velikosti 5x5 pixelů pro velikost markeru 6x6 bitů (a to bez ohledu na okraj markeru). Aby se zabránilo deformaci musí být tento parametr úměrný počtů bitů a velikosti okraje nebo alespoň mnohem vyšší než velikost markeru.
- čtvrtý parametr je proměnná výstupního obrazu,
- poslední parametr je volitelný, určuje šířku černých okrajů markeru. Zadaná velikost musí být úměrná počtu bitů. Například hodnota 2 znamená, že hranice bude mít šířku ekvivalentní velikosti dvou vnitřních bitů. Výchozí hodnota je 1.

4 Detekce

Proces detekce markerů se skládá z několika kroků:

1. převod vstupního obrázku na černobílý obrázek nebo na obrázek barevného modelu RGB,
2. provést binární thresholding (prahování),
3. provést detekci kontur,
4. nalézt kandidáty na markery,
5. extrahovat bity z markeru,
6. umístit marker do prostoru [21].

Výhodou OpenCV knihovny je, že během implementace se nemusí dělat všechny kroky. Protože knihovna provádí kroky sama. tzn. nemusí je provádět programátor.

4.1 Detekce markerů

Každý detekovaný marker obsahuje:

- umístění 4 rohů v obraze,
- ID markerů.

DetectMarkers() je funkcí pro detekci markerů. Ve výpisu 2 je ukázka implementace kódu pro detekci markerů.

```
cv::VideoCapture inputVideo;
inputVideo.open(0);
cv::Ptr<cv::aruco::Dictionary> dictionary = cv::aruco::
    getPredefinedDictionary(cv::aruco::DICT_6X6_50);
while (inputVideo.grab()) {
   cv::Mat image, imageCopy;
   inputVideo.retrieve(image);
   image.copyTo(imageCopy);
   std::vector<int> ids;
   std::vector<std::vector<cv::Point2f> > corners;
   cv::aruco::detectMarkers(image, dictionary, corners, ids);
   if (ids.size() > 0)
      cv::aruco::drawDetectedMarkers(imageCopy, corners, ids);
   cv::imshow("out", imageCopy);
   char key = (char) cv::waitKey(waitTime);
```

```
    if (key == 27)
       break;
}
```

Výpis 2: Detekce markeru

Funkce má tyto parametry:

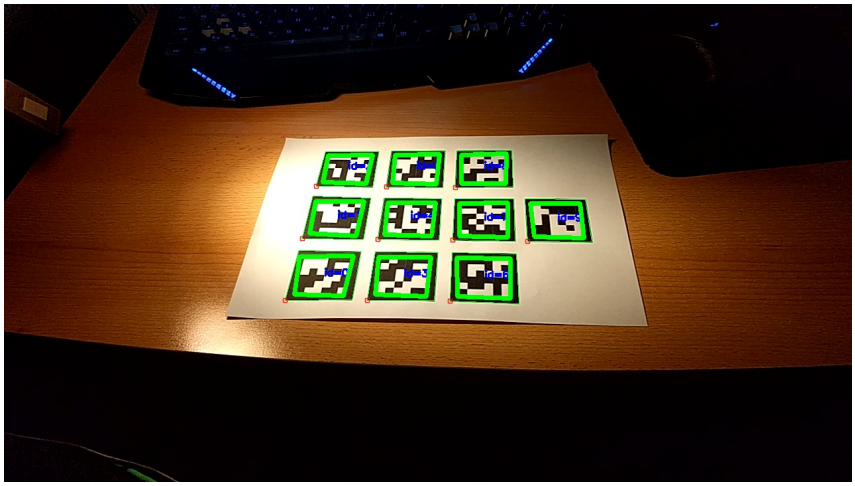
- prvním parametrem je obrázek, který bude detekován. V případě mé aplikaci to jsou snímky z kamery.
- druhým parametrem je slovník,
- z detekovaných markerů jsou uloženy rohy markeru a ID markeru do proměnných vektorů corners a ids,
- čtvrtým parametrem jsou parametry, které lze modifikovat během prováděné detekce,
- posledním parametrem jsou kandidáti na markery, parametr vrací seznam všech kandidátů na markery.

Poslední dva parametry jsou nepovinné, proto nejsou použity v ukázkovém kódu.

Vektor corners je seznam všech detekovaných rohů z markerů. Každý marker má 4 rohy, které jsou vráceny v určitém pořadí (prvním vraceným rohem je levý horní roh, a poté rohy ve směru hodinových ručiček). Jeden z parametrů funkce detectMarkers() je DetectorParametrs objekt. Tento objekt obsahuje všechny možnosti, které mohou být upravovány během detekce. Tyto parametry jsou popsány v kapitolách:

- Thresholding 4.3
- Detekce obrysů 4.4
- Bitová extrakce 4.6

Po detekci markeru se musí ověřit zda byl marker detekován, a to funkcí drawDetectedMarkers(), která vykreslí kolem všech markerů čtverec s identifikačním číslem markeru, jak je možné vidět na obrázku 9.



Obrázek 9: Obrázek s detekovanými markery

4.2 Převod barvy vstupního obrázku

Před samostatnou detekci se nejdříve musí zpracovat vstupní obraz z kamery, aby odpovídal formátu, se kterými algoritmy pracují. Protože se předpokládá, že se technik bude pohybovat kolem automobilu v prostředí s nerovnoměrným osvětlením, musí se načtený snímek převést na černobílý obrázek nebo na obrázek barevného modelu RGB.

Během těchto převodů jsem zjistil, že nelze převádět černobilý obrázek zpět do barevného obrázku. Proto v aplikaci převádím vstupní obrázek na barevný model RGB, protože bylo zapotřebí, aby výsledný obrázek s detekovaným markerem byl opět převeden do modelu RGBA.

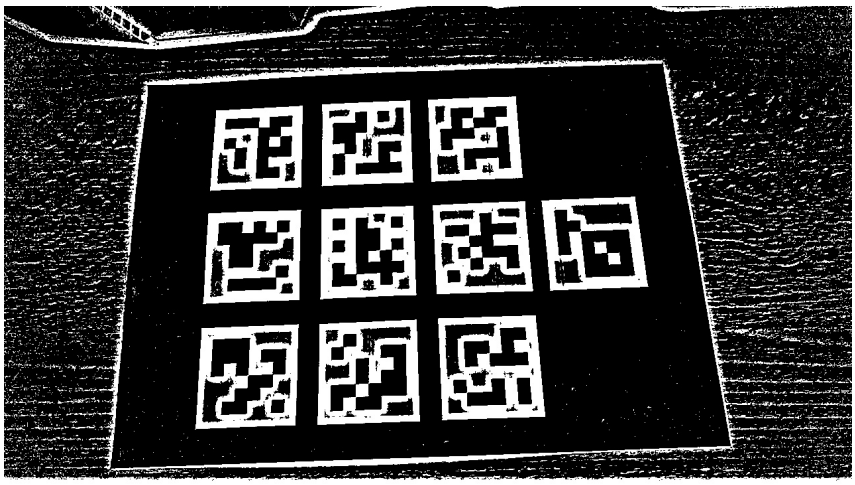
V OpenCV je na převod barvy obrázku funkce cvtColor(), která potřebuje tyto parametry:

- matici vstupního obrázku inputImage,
- matici výstupního obrázku outputImage,
- Code -kód konverze barvy obrázku.

Jako kód konverze byl použit COLOR_RGBA2BGR z OpenCV knihovny, který převádí snímky z modelu RGBA do modelu RGB. Po detekci markeru bylo zapotřebí použít kód COLOR_BGR2RGBA z OpenCV knihovny, který převádí snímky z modelu RGB na model RGBA.

4.3 Thresholding

Funkce thresholding transformuje každý pixel vstupního snímku na černé nebo bílé pixely. Tento krok je důležitý provést před vyhledávním kontur z markeru. Pro binarizaci obrazu je možné použít adaptivní thresholding (adaptnivní prahovnání), který je zobrazen na obrázku 10. Metoda počítá práh pro malé části obrazu místo globálnímu nastavení prahu pro celý snímek.



Obrázek 10: Prahování obrázku

Thresholding může být přizpůsoben podle uživatele v následujících parametrech:

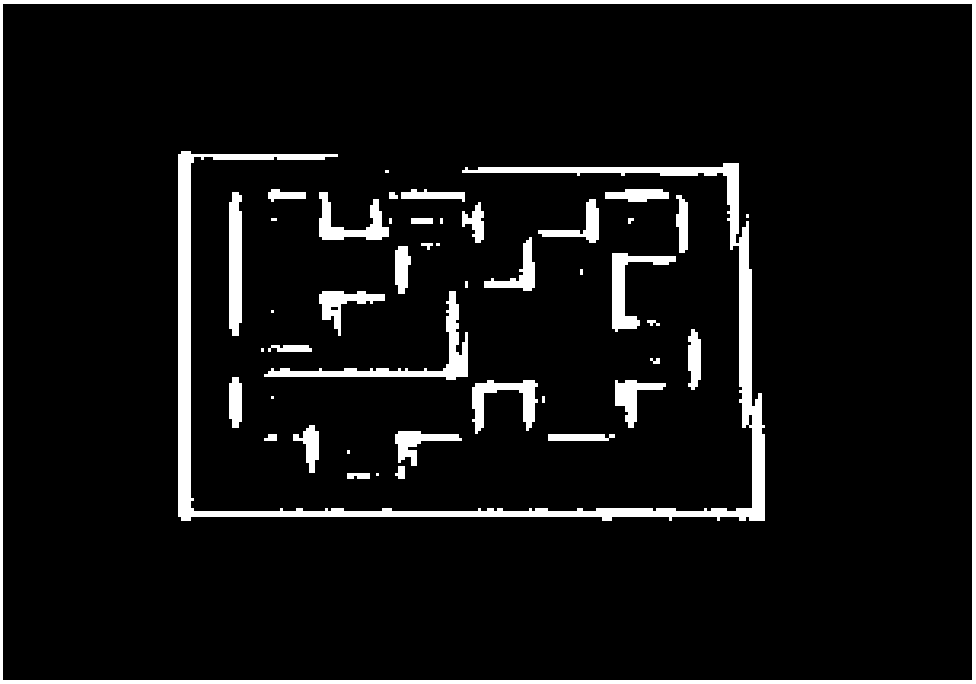
- adaptiveThreshWinSizeMin,
- adaptiveThreshWinSizeMax,
- adaptiveThreshWinSizeStep.

Pro funkci adaptivního prahování jsou zvoleny části velikosti obrazů v pixelech z intervalu od adaptiveThreshWinSizeMin do adaptiveThreshWinSizeMax. Parametr adaptiveThreshWinSizeStep indikuje navyšení velikosti části obrazu z hodnot adaptiveThreshWinSizeMin a adaptiveThreshWinSizeMax. Po nastavení hodnot podle výpisu 3, budou velikosti části obrázu 5,9,13,17 a 21.

```
int adaptiveThreshWinSizeMin =5;
int adaptiveThreshWinSizeMax =21;
int adaptiveThreshWinSizeStep =4;
```

Výpis 3: Nastavení parametrů

Během nastavování hodnot parametrů je potřeba si dávat pozor na to, aby se nenastavovaly nízké hodnoty pro části obrazu. Pokud má marker velkou velikost a je nastavená malá velikost části obrazu, může dojít k deformaci rámečku markeru a marker nebude detekovaný. Jak je zobrazeno na obrázku 11.



Obrázek 11: Deformace obrázků [22]

4.4 Detekce obrysů

Po adaptovaném prahování (adaptive thresholding) následuje krok detekce obrysů z markerů. Všechny obrysy kandidátů, které nepřípomínají kandidáta na marker jsou během detekce odfiltrovány.

Parametry minMarkerPerimeterRate a maxMarkerPerimeterRate determinují minimální a maximální velikost markeru, tzn. obvod markeru. Nejsou specifikovány v hodnotách absolutních pixelů - místo toho jsou určeny vzhledem k maximální dimenzi vstupního obrazu. Vstupní obraz je fotka nebo snímek z kamery či videa. Například pokud je obrázek o velikosti 640x480 pixelů a relativní obvod markeru je na hodnotě 0.05, povede to k minimálnímu obvodu okraje markeru $640 * 0.05 = 32$ pixelů, protože 640 pixelů je maximální rozměr obrázku. Vzoreček pro výpočet velikosti minimálního obvodu markeru:

$$x * y = minMarkerPerimeterRate$$

Kde:

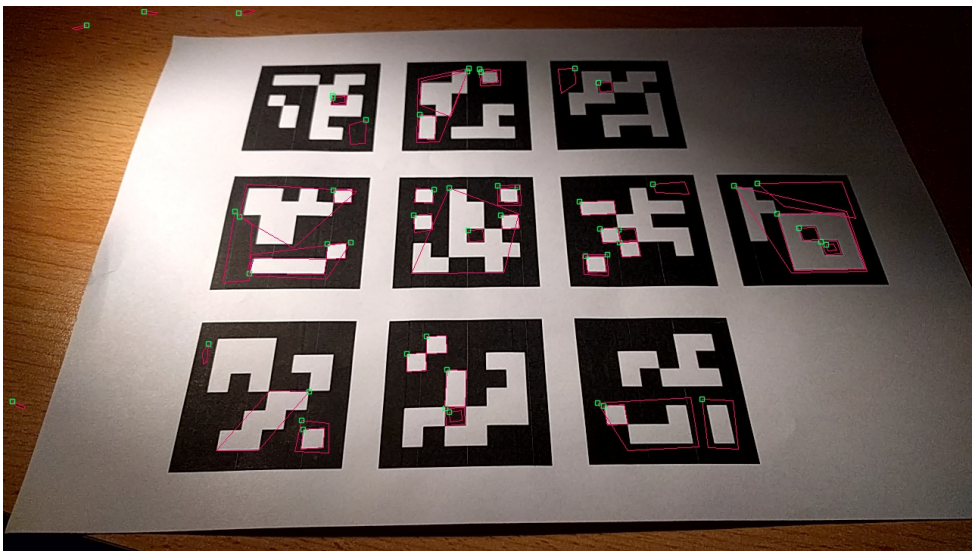
- x je maximální velikost obrázku,
- y je relativní obvod markeru.

Pokud má parametr minMarkerPerimeterRate nastavenou nízkou hodnotu může výrazně ovlivnit výkon detekce, protože v příštích krocích bude brát větší ohled na malé obrysy. Tento problém nebude nijak ovlivněn parametrem maxMarkerPerimeterRate, protože je v obrázku

pořád více malých obrysů než těch velkých. Jestli-že se nastaví minMarkerPerimeterRate na hodnotu 0 maxMarkerPerimeterRate hodnotu 4 nebo více, budou se detekovat všechny obrysy v obrázku. Toto nastavení není vhodné kvůli zhoršenému výkonu filtrování obrysů.

4.5 Kandidáti na markery

Po nalezení obrysů se provádí fáze přiblížení polygonu. Funkce approxPolyDP() je určena k přibližování polygonální křivky s určenou přesností, tzn. funkce se blíží ke křivce nebo mnohouhelníku s jinou křivkou nebo polygonem s menšími vrcholy. Vzdálenost mezi nimi je menší nebo rovná určené přesnosti.
Pro každého kandidáta je použita polygonální aproximace a jsou přijmáni pouze ti kandidáti, kteří jsou podobní čtvercovému tvaru. Na obrázku 12 jsou zobrazeny kandidáti na markery.

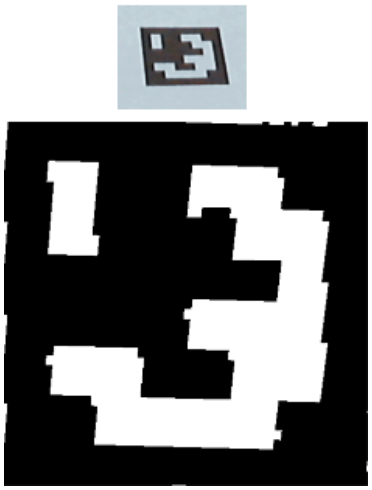


Obrázek 12: Kandidáti na markery

Hodnota parametru PolygonalApproxAccuracyRate určuje maximální chybovost, kterou může polygonální aproximace produkovat. Tento parametr odpovídá délce obvodu markeru kandidáta v pixelech.

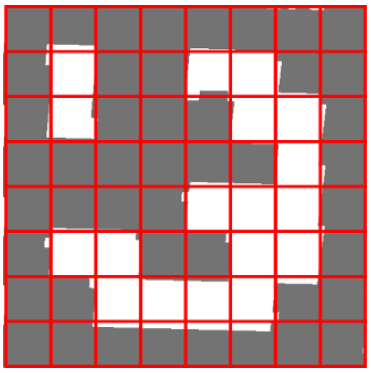
4.6 Bitová extrakce

Po detekci kandidátů je analyzován každý bit každého kandidáta v pořadí, aby se zjistilo zda je to marker nebo ne. Před analýzou samostatného binárního kódu je zapotřebí extrahovat bity. Proto aby marker byl narovnán do do roviny fotoaparátu, musí se provést perspektivní transformace, ta odstraní perspektivní zkreslení. Následně použitím metody threshold Otsu funkce odstraní šedé pixely z markeru, aby zůstaly pouze černé a bílé pixely. Na obrázku 13 je vidět jak vypadá marker, když se odstraní perspektivní zkreslení.



Obrázek 13: Perspektivní odstranění [23]

Poté je obrázek rozdělen na pravidelnou mřížku jak je zobrazeno na obrázku 14. Každé buňce je přiřazena hodnota 0 nebo 1 v závislosti na bílem či černém pixelu. Pokud je na markeru bílé políčko je v buňce bitová 1, pokud je černé políčko je v buňce bitová 0.



Obrázek 14: Mřížka na markeru [24]

Zde jsou parametry, které si může uživatel nastavit:

- markerBorderBits,
- minOtsuStdDev,
- perpectiveRemovePixelPerCell,
- perspectiveRemoveIgnoredMarginPerCell,

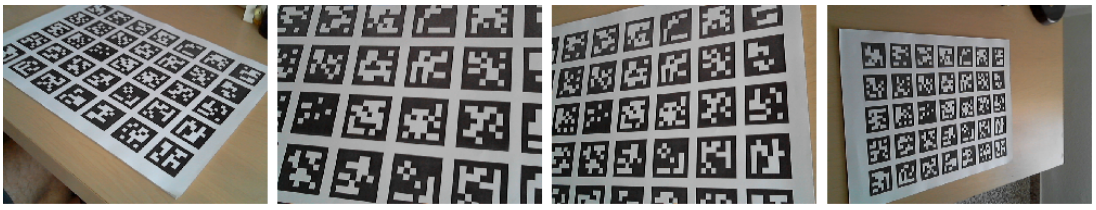
Parametr markerBorderBits určuje jak velkou šířku okraje bude mít marker. Hodnota 2 značí, že okraj bude mít šířku dvou vnitřních bitů.

Parametr minOtsuStdDev určuje minimální směrodatnou odchylku pixelů pro provedení Otsu funkce. Pokud je odchylka nízká znamená to, že všechny čtverce jsou buď bílé nebo černé. V takovém případě se nemusí používat funkce Otsu, ale místo toho se všechny bity nastaví na hodnotu 0 nebo 1 v závisloti na tom, zda je odchylka vyšší nebo nižší než 128.

Parametr perpectiveRemovePixelPerCell určuje počet pixeů (v každé buňce) po odstranění perspektivního vyhlazení. Například je vybrán marker 6x6 bitů o velikosti okraje 1 (markerBorderBits) a je zapotřebí spočítat kolik pixelů/bitů zůstane po odstranění perspektivního vyhlazení. Příklad $6 + 2 * 1 = 8$ (okraj markeru bude započítán dvakrát) takže cekový počet buňěk bude 8x8.

4.7 Kalibrace kamery

Kalibrace kamery se provádí pouze jednou před detekcí markerů, následně se již provádět nemusí. Dalším krokem před detekcí markerů je zapotřebí zjistit z markeru odhad kamery v prostoru. Aruco knihovna tuto kalibraci již má zabudovanou pomocí funkce calibrateCamera(). Tato funkce uloží soubor s informacemi, které byly pořízené z kamery jako matici 3x3 prvků s ohniskovou vzdáleností a koeficientem zkreslení. Kalibrace kamery se provádí, tak že se na papír vytiskne "šachovnice markerů", a poté se "šachovnice" snímá fotoaparátem ze všech stran, jak je patrné z obrázku 15.



Obrázek 15: Aruco šachovnice markerů [25]

Funkce vrací tyto hodnoty:

- matici kamery 3x3 prvků s ohniskovou vzdáleností a souřadnicemi středu kamery, tzn. Vnitřní parametry,
- koeficienty zkreslení, tzn. Vektor s pěti nebo více prvky, který modeluje zkreslení způsobené fotoaparátem.

Tyto hodnoty se vloží do souboru a uloží se do interní paměti mobilního telefonu, nebo počítače. Ve výpise 4 je napsán kód pro kalibraci kamery:

```
Mat cameraMatrix, distCoeffs;
vector< Mat > rvecs, tvecs;

Ptr<Dictionary> dictionary = getPredefinedDictionary(
    PREDEFINED_DICTIONARY_NAME::DICT_6X6_50);
Ptr<aruco::DetectorParameters> detectorParams = aruco::DetectorParameters
    ::create();

Ptr<aruco::GridBoard> gridboard =GridBoard::create(5, 7, 0.05, 0.01,
    dictionary);
Ptr<aruco::Board> board = gridboard.staticCast<aruco::Board>();

aruco::calibrateCameraAruco(allCornersConcatenated, allIdsConcatenated,
    markerCounterPerFrame, board, imgSize, cameraMatrix, distCoeffs, rvecs
    , tvecs, 0);

saveCameraParams(outPutFile, imgSize, 1, 0, cameraMatrix, distCoeffs,
    repError);
```

Výpis 4: Kalibrace kamery

Důležité proměnné:

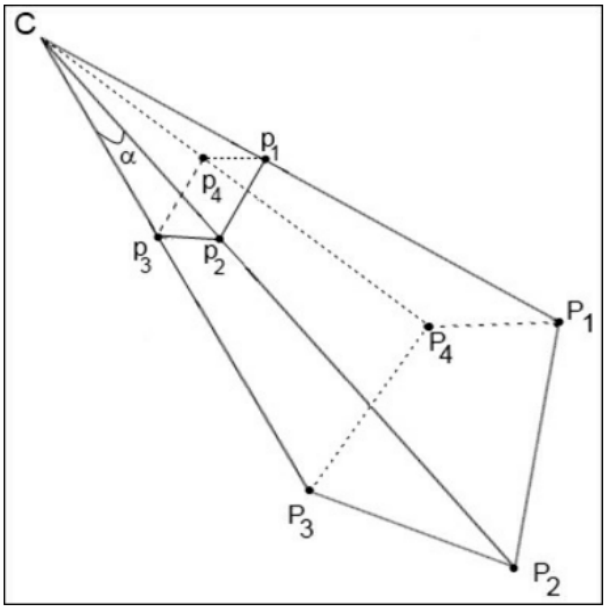
- rvecs je vektor rotace tzn. podle toho jak je otáčena kamera okolo markeru se ukládají informace o rotaci markeru do této matice,
- tvecs je transformační matice, která popisuje převod souřadnic bodů v trojrozměrném prostoru ze světových souřadnic do souřadnic kamery [26],
- cameraMatrix výstupní matice 3x3 prvků s ohniskovou vzdálenosti

$$A = \begin{bmatrix} f_x & 0 & 0 \\ 0 & f_y & 0 \\ c_x & c_y & 1 \end{bmatrix}$$

  Kde f_x, f_y jsou vzdálenosti ohniska v pixelech a c_x, c_y jsou souřadnice bodu v pixelech, který bývá ve středu obrazu,
- distCoeffs výstupní koeficientu zkreslení o čtyřech, pěti, osmi, dvanácti nebo čtrnácti prvcích.

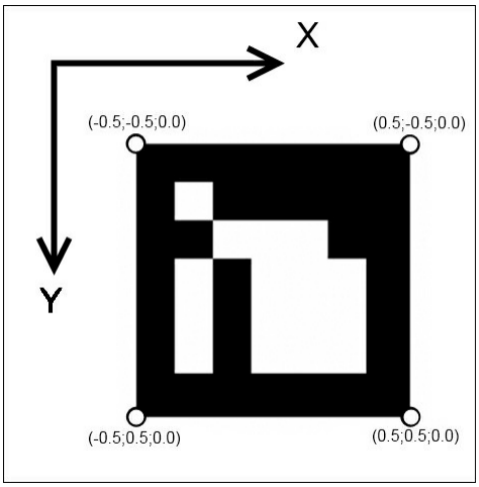
4.8 Odhad Kamery markeru v prostoru

Jelikož jsou známy přesná umístění rohů markeru, odhad transformace mezi kamerou a markerem v 3D prostoru nebude problém. Pro odhad kamery se používá proces Eukleidovské transformace, ta k odhadu potřebuje pouze matici rotace (rvecs) a transformační matici (tvecs). Obě tyto matice jsou popsány v kapitole Kalibrace kamery 4.7 Matice rotace a transformační matice jsou již známé z markeru. C označuje střed fotoaparátu. Body P1-P4 jsou souřadnice bodů v



Obrázek 16: Euklidová transformace [27]

trojrozměrném prostoru světových souřadnic a body p1-p4 jsou jejich projekce v rovině snímku fotoaparátu. Pomocí matice vnitřních parametrů a bodových projekcí na rovině snímku (P1-P4), lze najít relativní transformaci mezi fotoaparátem a pozici markeru v reálném světě (p1-p4). Ale jak dostat souřadnice markeru v reálném prostředí? Jelikož marker má vždy čtvercovou podobu a všechny vrcholy jsou umístěné v jedné rovině dají se definovat jejich rohy takto (obrázek 17). Na obrázku 17 je marker umístěn do roviny XY a střed markeru odpovídá bodům (0.0, 0.0, 0.0) V tomto případě budou souřadnice uprostřed markeru (osa z je kolmá).

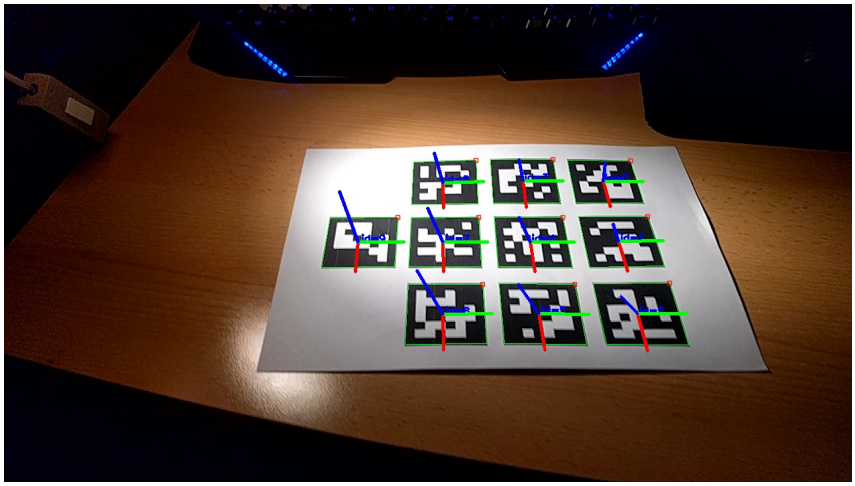


Obrázek 17: Definice rohů markeru [27]

Aruco poskytuje funkci estimatePoseSingleMarkers() pro odhad umístění všech detekovaných markerů v reálném prostředí, zde budou popsány všechny jeho parametry:

- první parametr je vector corners, seznam všech rohů detekovaných z markerů podrobněji napsáno v kapitole Detekce markerů 4.1,
- druhý parametr je velikost markeru v metrech nebo jiné jednotce,
- dalšími parametry jsou cameraMatrix, distCoeffs, rvecs a tvecs tyto parametry jsou popsány v kapitole Kalibrace kamery 4.7.

Na obrázku 18 je zobrazeno jak funkce vykreslí doprostřed markeru osy X: červená, Y: zelená, Z: modrá, toto je souřadnicový systém markeru.



Obrázek 18: Vykreslené osy

4.9 Výběr slovníku

Aruco obsahuje Slovníkové třídy, které reprezentují slovníky markeru. Součástí slovníku je kromě velikosti markeru a čísla také číslo udávající mezery mezi markery. Tyto mezery určují minimální vzdálenost mezi markery a určují schopnost detekce chyb a korekce slovníku.

Obecně platí, čím menší velikost slovníku a větší velikost markeru - tím se zvyšuje mezera mezi markery a naopak. Detekce markeru s vyššími velikostmi je složitější vzhledem k vyššímu počtu bitů a jejich extrakce z markeru.

5 Aplikace v praxi

Detekce markerů může být využita ve spoustě lidských činností, od údržby automobilu až po výukové účely anatomie těla. Já jsem pro svou bakalářskou práci vybral činnost údržby automobilu konkrétně dolití vody do nádržky od ostřikovačů. Aplikace slouží jako návod pro technika a měla by pomocí detekce markerů usnadnit práci technika, který bude vykonávat tyto úkony. V jednotlivých kapitolách níže popisuju funkčnost, vzhled aplikace a popis úkonu technika. Umístění markerů a průběh testování budou popsány v kapitole 6.

Aplikace byla vytvořena pro mobilní telefony s architekturou procesoru abi-arm64-v8a, jelikož OpenCV má knihovny kompatibilní s touto architekturou. Aplikaci bohužel nelze nainstalovat na mobilním zařízení s architekturou procesoru armeabi-v7a.

5.1 Seznámení s aplikací

Aplikace by měla popsat jednotlivé kroky technika, který by měl vyměnit vodu v ostřikovačích. Technik by nejdříve měl najít dveře od řidiče pomocí markeru. Po nalezení dveří by měl otevřít dveře od řidiče a najít marker pro páku na otevření kapoty. Následně by měl zatáhnout za páku pro otevření kapoty a přemístit se ke kapotě, kterou otevře. Technik by měl hned na první pohled nalézt nádržku na vodu do ostřikovačů pomocí markeru. Následně by měl nalézt lahev s vodou do ostřikovačů. Po nalezení láhve s vodou do ostřikovačů by se technik měl vrátit ke kapotě vozu a měl by nalít vodu do nádržky. Daná aplikace byla vyvíjena pro mobilní telefony se systémem Android. Pro testovací účely bakalářské práce byl použit mobilní telefon značky LG Nexus 5x. Před prvním spuštěním aplikace je zapotřebí potvrdit práva pro použití kamery a externí paměti telefonu v nastavení telefonu. Po spuštění aplikace se uloží do interní paměti mobilního telefonu soubory:

- soubor vytvořený z kalibrace kamery,
- soubor s parametrama pro detekci markerů
- soubor s úkonama pro technika.

Jednotlivé úkony z JSON souboru se ukážou v aktivitě "Technické úkony", které bude uživatel vykonávat.

5.2 Popis aplikace

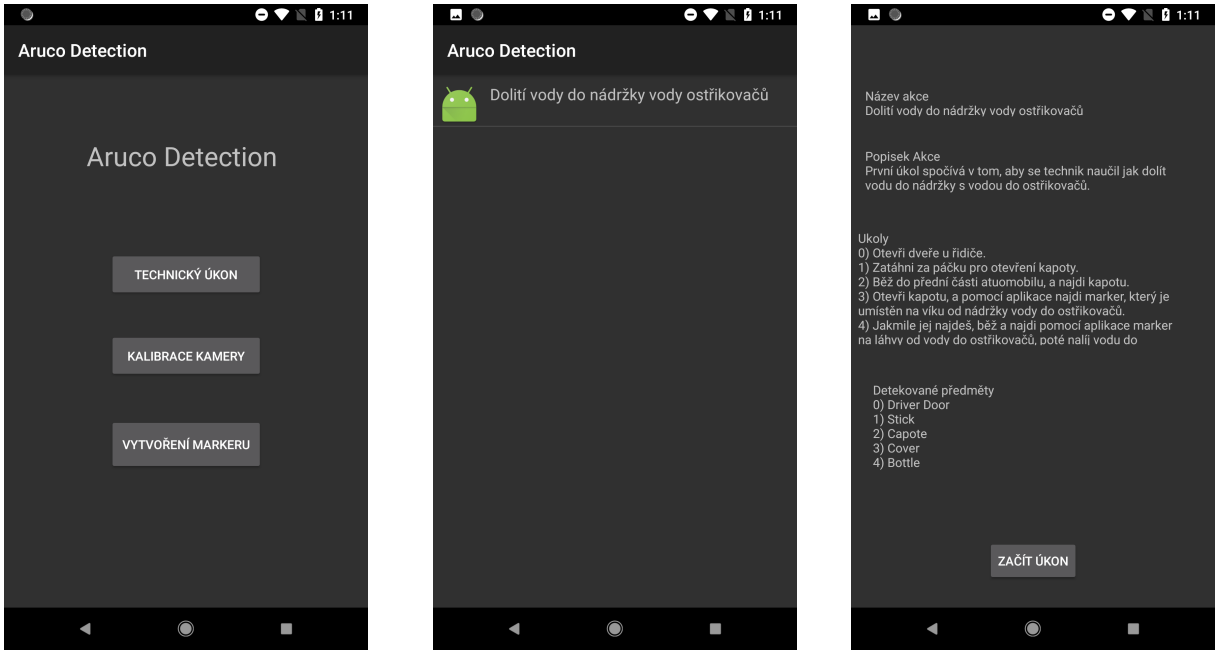
V této kapitole bude popsán vzhled a funkčnost aplikace. Po spuštění aplikace uživatel uvidí na hlavní obrazovce 3 tlačítka jak je možné vidět na obrázku 19a.

1. první tlačítko "Technický úkon" je zadání úkonu,
2. druhé tlačítko "Kalibrace kamery" je pro kalibraci kamery,

3. třetí tlačítko "Vytvoření markeru" slouží pro rychlé vygenerování deseti markerů ze slovníku 6x6_50.

Na obrázku 19b je možné vidět seznam všech technických úkonů. Technik si vybere ze seznamu úkonů a přejde do vybraného detailu úkonu. Jak můžete vidět na obrázku 19c obrazovka je rozdělena do několika sekcí:

- nahoře je název akce,
- pod názvem je popisek akce,
- pod popiskem jsou úkoly pro technika,
- poslední položkou jsou detekované předměty, tyto předměty bude technik hledat popřípadě používat během úkonu.

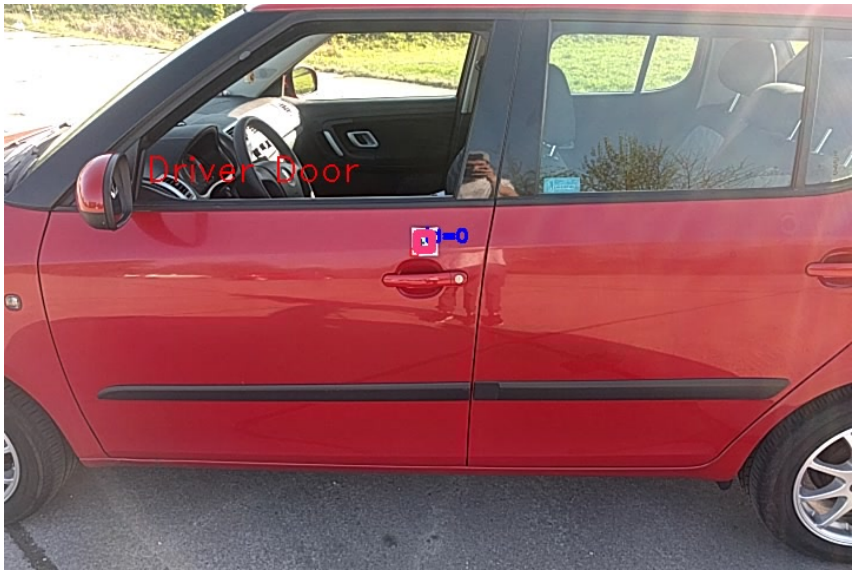


(a) Menu aplikace (b) Seznam úkonů (c) Detail úkonu

Obrázek 19: Rozhraní aplikace

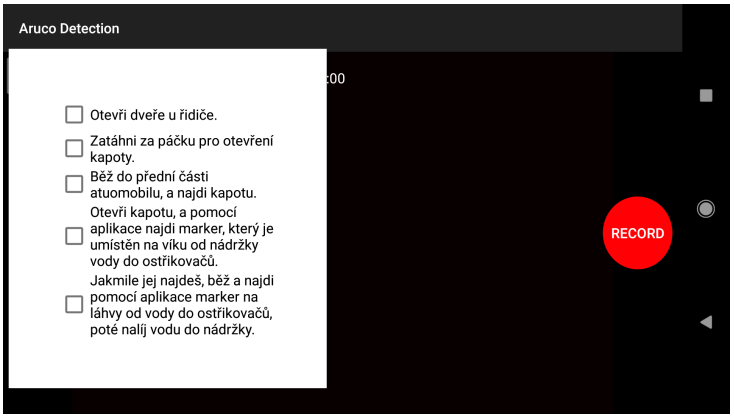
Předměty na, kterých je marker umístěn musí být napsány v Anglickém jazyce, to kvůli popiskům k markerům, které se zobrazují vedle markeru, jelikož aruco neumí pracovat s UTF-8 kódování.

Pokud technik porozumí instrukcím klikne na tlačítko "Začít úkon". Aplikace ho přepne do aktivity pro detekci markeru. Technik namíří fotoaparátem telefonu na marker, objeví se na obrazovce název předmětu vedle markeru, jak je možné vidět na obrázku 20



Obrázek 20: Dveře řidiče s markerem id 0

Po rozkliknutí tlačítka "Úkoly" uvidí technik seznam instrukcí, které musí udělat, to je zobrazeno na obrázku 21. Instrukce jsou seřazené podle úkolů z aktivity "Detail úkolu", také jsou tyto úkoly spojeny s markerami.



Obrázek 21: Seznam úkolů z aktivity detail

Po vykonání úkolů odškrtne tento úkol a marker změní barvu z červené na zelenou. Zelená barva markeru signalizuje technikovi, že tento marker je již vykonán tzn. úkol je odškrtnut v seznamu úkolů a tímto potvrdil vykonání úkolu. Každý marker má své ID, toto ID je propojeno s úkolem a předmětem pro použití.

Po kliknutí na kalibraci kamery se technik dostane na obrazovku se vstupem z kamery. Během kalibrace kamery musí technik namířit kameru mobilního fotoaparátu na šachovnici aruco markeru. Během detekce této šachovnice se uloží parametry kamery do souboru na interní úložiště.

6 Testování přesnosti a funkčnosti

Během testování detekce přesnosti a funkčnosti jsem testoval detekci na markerech 6x6_50. Pro svou bakalářskou práci vybral činnost údržby automobilu konkrétně dolití vody do nádržky od ostřikovačů, proto jsme nasimuloval prostředí ve kterém automechanik pracuje. Prostředí ve kterém automechanik pracuje:

- venkovní prostředí za dobrých světelných podmínek,
- venkovní prostředí za zhoršených světelných podmínek (zatažené počasí),
- vnitřní prostředí garáže.

Po těchto testech jsem udělal test s markery 6x6_250 za zhoršených světelných podmínek, abych mohl potvrdit tvrzení, že čím větší číslo slovníku tím je větší chybovost při prováděné detekci markerů.

Ve všech testech detekce jsem detekoval markery pomocí mobilního telefonu s Androidem značky LG Nexus 5X. K testování detekce markerů byl použit můj mobilní telefon LG Nexus 5x. Rozlišení kamery mobilního telefonu bylo nastaveno na 720x480 pixelů to kvůli zrychlení detekce markerů a výsledným videím, aby nebyly zpomalené při vyšším rozlišení. Na testování byl použit automobil značky Škoda Fábia červené barvy.

Všechny markery byly umístěny na stejné části kapoty. Byly použity markery 6x6_50 a 6x6_250 s velikosti 200x200 pixelů.

Markery byly umístěny na části automobilu podle předmětů v zadání úkonu. Umístění markerů:

- ID 0 umístěn na dveře řidiče,
- ID 1 umístěn na páku od otevření kapoty,
- ID 2 umístěn na kapotu,
- ID 3 umístěn na výko nádržky od ostřikovačů,
- ID 4 umístěn na láhev od vody do ostřikovačů.

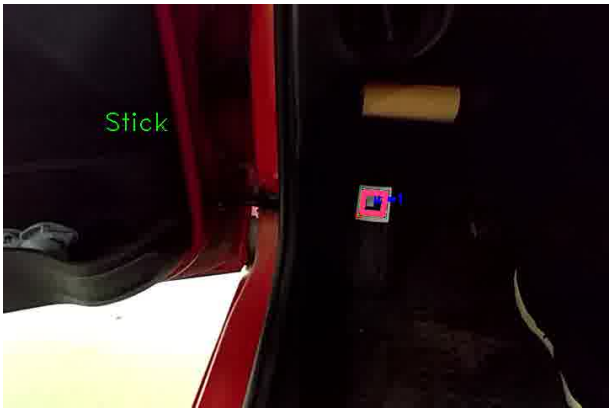
Průběh testování: Poté, co jsem umístil jednotlivé markery na části automobilu spustil jsem aplikaci v mobilním telefonu a vybral jsem z menu "Technický úkon". Následně se spustil seznam jednotlivých technických úkonů a byl vybrán úkon "Dolití vody do nádržky vody ostřikovačů". Aplikace přejde na detail úkonu. Seznam úkolů je číslován od 0 do n počtu úkolů. Po přečtení seznamu úkolů jak postupovat během úkonu, jsem kliknul na tlačítko "Začít úkon".

Mobilní telefon byl namířen na dveře od řidiče automobilu, to vyvolalo zčervenání markeru a objevení názvu předmětu a ID markeru. Poté jsem kliknul na červené tlačítko, které spustilo natáčení videa. Po kliknutí na tlačítko "Úkoly"jsem zjistil jaké úkoly mám vykonat. Přečetl jsem si nultý úkol "Otevři dveře u řidiče."a šel jsem otevřít dveře u řidiče, tyto dveře měli marker ID 0 (obrázek 22a). Následně jsem odškrtnul nultou položku ze seznamu. Marker změnil barvu z červené na zelenou, tzn. nultý úkol byl dokončen. Následně jsem si přečetl další úkol "Zatáhni za páčku pro otevření kapoty.", takže jsem hledal marker s ID 1 (obrázek 22b), který byl umístěn vedle páčky pro otevření kapoty. Zatáhnul jsem za páčku a odškrtnul jsem v seznamu úkolů první úkol.

Mým dalším úkolem bylo najít marker s ID 2, dle aplikace jsem měl udělat "Běž do přední části atuomobilu, a najdi kapotu.". Poté co jsem přišel k přední části automobilu a nalézl jsem marker s ID 2, (obrázek 22c), který byl umístěn na kapotě jsem odškrtnul úkol jako hotový. Přečetl jsem si další úkol "Otevři kapotu, a pomocí aplikace najdi marker, který je umístěn na víku od nádržky vody do ostřikovačů.". Po otevření kapoty automobilu jsem pomocí aplikace nalezl marker s ID 3, (obrázek 22d). Tento marker byl umístěn na víku od nádržky s vodou do odstřikovačů.

V seznamu úkolů jsem si odškrtnul třetí úkol jako splněný. Stejně jsem postupoval i u úkolu "Jakmile jej najdeš, běž a najdi láhev (na kterém je umístěn marker) od vody do ostřikovačů, poté nalíj vodu do nádržky."V posledním úkolu jsem musel najít láhev s vodou do ostřikovačů, která byla v kufru a měla na sobě daný marker s ID 4 (obrázek 22e). Nalezenou láhev jsem vzal a nalil vodu do nádržky. Na konci úkonu, kdy mám vše hotové jsem kliknul opět na červené tlačítko, kteréa zastavilo nahrávání videa.

(a) ID 0: umístěň na dveře řidiče



(b) ID 1: umístěn vledle páky pro otevření kapoty

(c) ID 2: umístěn na kapotu

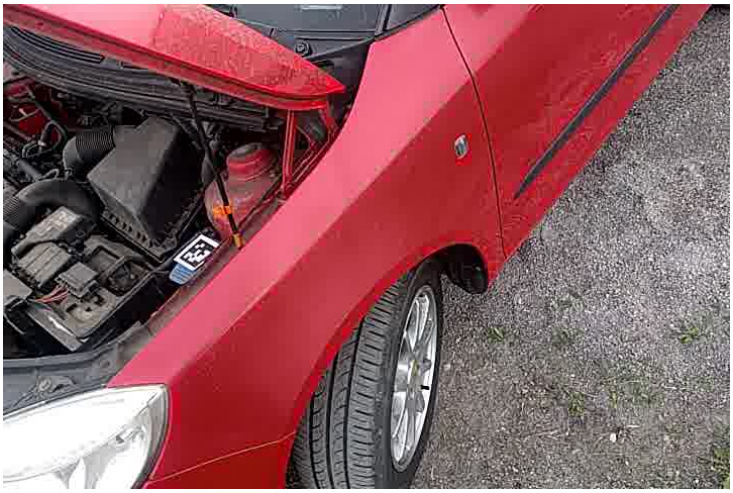
(d) ID 3: umístěn na víko nádržky

(e) ID 4: umístěn na láhev od vody do ostřikovačů

Obrázek 22: Umístění markeru na části vozidla

Takto jsem postupoval u všech testů. Ze všech natočených videí jsem pomocí programu extrahoval snímky do adresáře a s nimi pracoval při tvoření tabulky úspěšnosti, funčnosti jednotlivých markerů. Některé markery nejsou detekované z důvodu překrytí rohu markeru objektem nebo přesvětlením markeru, či špatnému zachycení markeru. Na obrázku 23 je vidět marker ID 3,

který je překrytý rámem karosérie.



Obrázek 23: Marker ID 3 překryt rámem karosérie

Popis sloupců v tabulkách:

- ID markeru: identifikační číslo markeru,
- detekované markery: jsou markery, které byly označeny červenou nebo zelenou barvou,
- nedetekované markery: jsou markery, které nebyly aplikací vůbec detekované tzn. nebyly označené barvou,
- celkový počet snímků: celkový počet snímků, kde byl viditelný marker,
- úspěšnost detekovaných markeru jsem počítal jako $\frac{detekovane}{celkovypocetsnimku} * 100 = uspesnost$.

6.1 Venkovní prostředí za dobrých světelných podmínek

Pro testování detekce markerů v případě venkovního prostředí za dobrých světelných podmínek jsem použil markery ze slovníku 6x6_50. Výsledky úspěšnosti detekce těchto markerů jsou uvedeny v tabulce 1.

ID markeru	Detekované	Nedetekované	Celkový počet snímků	Úspěšnost [%]
0	70	1	71	98.5
1	37	60	97	38.1
2	75	95	170	44.1
3	70	1	71	98.5
4	31	0	31	100
Celková úspěšnost	283	157	440	64.3

Tabulka 1: Tabulka úspěšnosti Markerů za dobrých světelných podmínek

Z tabulky 1 je vidět, že největší úspěšnost měl marker s ID 4 (obrázek 24a), kdy měl úspěšnost 100 procent. Co stojí za povšimnutí je také marker s ID 0 a 3, kdy tato úspěšnost u obou markeru byla 98.5 procent. Nejméně úspěšnými markery byly markery s ID 1 (obrázek 24b) a 2 kdy tato úspěšnost byla v průměru 41.1 procent. Celková úspěšnost z celkového počtu snímků 440 bylo detekovanuých 283 snímků a nedetekovaných 157 snímků. Celková úspěšnost detekce ve venkovním prostředí za optimálních podmínek byla 64.3 procenta.

(a) Marker ID 4 s největší úspěšnosti

(b) Marker ID 1 s nejmenší úspěšností

Obrázek 24: Největší úspěšnost a nejmenší úspěšnost detekce za dobrých světelných podmínkách

6.2 Venkovní prostředí za zhoršených světelných podmínek

Pro testování detekce markerů v případě venkovního prostředí za zhoršených podmínek jsem použil opět marker ze slovníku 6x6_50. Výsledky úspěšnosti detekce těchto markerů jsou uvedeny v tabulce 2.

ID markeru	Detekovaných	Nedetokovaných	Celkový počet snímků	Úspěšnost
0	115	53	168	68.5
1	60	0	60	100
2	49	33	82	59.8
3	63	10	73	86.3
4	56	15	71	78.9
Celková úspěšnost	343	111	454	75.6

Tabulka 2: Tabulka úspěšnosti testu za zhoršených světelných podmínek

Stoprocentí úspěšnost měl marker ID 1 (obrázek 25a), který byl umístěn na nádobce ostřikovačů. Marker s ID 3 měl po detekci úspěšnost 86.3 procent. Z celkového počtu 71 snímků byl marker s ID 4 detekován na 56 snímcích. Nejméně úspěšný marker byl marker s ID 2 (obrázek 25b) , který měl z celkového počtu 82 snímků pouze detekovaných 49. Celková úspěšnost byla 75.6 procent, z celkového počtu 454 snímků bylo detekováno 343.

(a) Marker ID 1 s největší úspěšností

(b) Marker ID 2 s nejmenší úspěšností

Obrázek 25: Největší úspěšnost a nejmenší úspěšnost detekce za zhoršených světelných podmínkách

6.3 Markery ze slovníku 6x6_250

Testování detekce markerů v případě markerů ze slovníku 6x6_250, byly testovány venku za zhoršených světelných podmínek. Úspěšnost testování je popsána v tabulce 3

ID markeru	Detekovaných	Nedetokovaných	Celkový počet snímků	Úspěšnost
0	28	82	110	25.5
1	32	50	82	39.0
2	10	144	154	6.5
3	30	104	134	22.4
4	37	8	45	82.2
Celková úspěšnost	137	388	525	26.1

Tabulka 3: Tabulka úspěšnosti testu markerů ze slovníku 6x6_250

Nejmenší úspěšnost má marker s ID 2 (obrázek 26b), který byl umístěn na přední kapotě automobilu. Marker ID 1 s úspěšností 39 procent z celkového počtu snímků 82 má detekovaných 32. Marker s největší úspěšností ze všech detekovaných markerů je marker s ID 4 (obrázek 26a), který byl umístěn na láhvi od vody do ostřikovačů. Úspěšnost tohoto markeru je 82.2 procent. Celková úspěšnost byla 26.1 procent.

(a) Marker ID 4 s největší úspěšností

(b) Marker ID 2 s nejmenší úspěšností

Obrázek 26: Největší úspěšnost a nejmenší úspěšnost detekce markerů ze slovníku 6x6_250

6.4 Vnitřní prostředí garáže

Pro testování detekce markerů ve vnitřním prostředí, v mém případě garáži, jsem použil marker ze slovníku 6x6_50. Výsledky úspěšnosti detekce těchto markerů jsou uvedeny v tabulce 4.

ID markeru	Detekovaných	Nedetokovaných	Celkový počet snímků	Úspěšnost
0	84	107	191	44.0
1	101	11	112	90.2
2	64	36	100	64
3	72	27	99	72.7
4	85	1	86	98.8
Celková úspěšnost	406	182	588	69.0

Tabulka 4: Tabulka úspěšnosti testu markerů ve vnitřních prostorách garáže

Nejúspěšnějším markerem v případě vnitřního prostředí byl marker s ID 4 (obrázek 27a), který měl úspěšnost 98,8 procent. Druhým nejúspěšnějším markerem byl marker s ID 1, který z celkového počtu 112 snímků měl detekovaných 101. Nejhorší hodnoty úspěšnosti měl marker s ID 0 (obrázek 27b), kdy jeho úspěšnost činila pouhých 44 procent. Celková úspěšnost detekce markerů v případě vnitřního prostředí - garáže, byla 69 procent.

(a) Marker ID 4 s největší úspěšností

(b) Marker ID 0 s nejmenší úspěšností

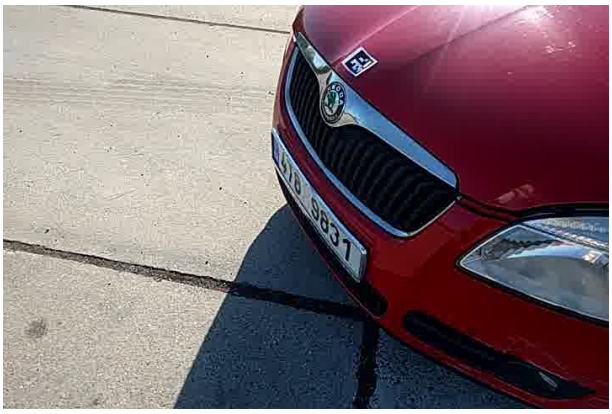
Obrázek 27: Největší úspěšnost a nejmenší úspěšnost detekce markerů vnitřního prostředí

6.5 Zhodnocení testování

Malá úspěšnost detekce markerů při tomto testování mohla být zapříčiněna tím, že jsem během detekce markerů byl v pohybu kolem auta. Kvůli rozmazaní jednotlivých snímků, nebyly detekovány markery. Nedetekování markerů bylo zapříčiněno:

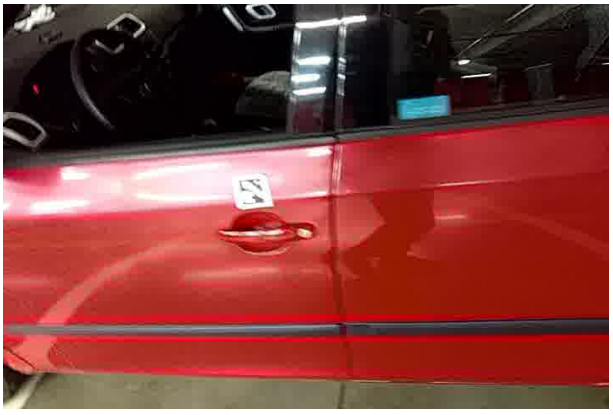
- rychlým pohybem mobilního telefonu během detekce markeru došlo k rozmázání snímaného snímku (obrázek 28a),
- kamera byla vůči markeru natočena ve špatném úhlu (obrázek 28b),
- kamera detekovala marker z velké dálky (obrázek 28c),
- přesvětelení markeru (obrázek 28d).

(a) Rozmazaný marker



(b) Marker byl ve špatném úhlu kamery

(c) Umístění kamery bylo vůči markeru daleko



(d) přesvětlení markeru.

Obrázek 28: Případy nedetekovaných markerů.

Nejúspěšnějším testem bylo testování detekce markerů ve venkonvím prostředí za zhoršených světelných podmínek na parkovišti.

Tyto markery byly ze slovníku 6x6_50, jak již bylo popsáno v tabulce 2 úspěšnost detekce markerů byla 75.6 procent. Tato úspěšnost, může být zapříčiněna tím, že markery neovlivňoval svit slunce. Nejůspěšnějšími markery byly markery, které byly umístěny uvnitř vozu. V tomto případě se jednalo o marker, který byl umístěn vedle páčky na otevření kapoty, a marker který byl umístěn na víku nádržky od vody do ostřikovačů.

Druhým úspěšným testem detekce markerů byl test vnitřního prostředí, kde celková úspěšnost byla 69 procent. Opět nejúspěšnějšími markery byly markery uvnitř vozu. Nejméně detekovaný marker byl marker, který byl umístěn na dveřích a kapotě vozu. Tuto špatnou detekci obou markerů mohl zapříčinit odraz světel v karosérii vozu, jak jsem popsal výše.

Třetím úspěšným testem detekce markerů byl test venkovního prostředí za dobrých světelných podmínek, kdy úspěšnost detekovaných markerů byla 64.3 procent. Nejúspěšnějšími detekovanými markery byly markery, které byly umístěny na dveřích od řidiče, víku od nádržky a na láhvy, která byla umístěna v kufru automobilu. Všechny tyto markery byly umístěny ve stínu. V případě dveří u řidiče to bylo zapříčiněno tím, že slunce svítilo na druhou polovinu atuobomilu, čímž vytvořil stín a marker byl lépe detekován.

Nejméně úspěšným testem detekce markerů byl test, který používal markery ze slovníku 6x6_250. Tento test byl proveden za zhoršených světelných podmínek. Při tomto testování jsem chtěl potvrdit že nejhůře detekovanými markery jsou markery ze slovníku, které mají vyšší počet markerů než je potřeba pro detekci. I při tak malé úspěšnosti 26.1 procent bylo prokázano, že nejlépe detekovanými markery byly markery, které se nacházely uvnitř vozu.

Jak jsem již popisoval jednotlivé testování, je jasné že nejlepší detekcí markeru mají markery, které se nacházejí uvnitř vozu, tzn. nacházejí se mimo okolní vlivy světla. Nejlepší umístění vozidla pro detekci markerů by bylo umístit vozidlo mimo dosah slunece či, mít umístěné světlo nad střechou vozu, tak aby světlo (sluneční svit) svítilo mimo dosah jednotlivých detekovaných markerů. V případě přesvětlených markerů by byla možnost implementace filtrů pro zpracování obrazu, tak aby filtr vyfiltroval přebytečné světlo z markerů.

Testováním detekce markerů ze slovníku 6x6_250 bylo potvrzeno tvrzení, že pro malé množství detekovaných markerů je dobré použít markery z menšího slovníku.

7 Závěr

Cílém bakalářské práce bylo seznámení se se základními pojmy v oblasti detekce markerů za pomocí obrazů. Vytvoření vlastní aplikace pro mobilní telefon s operačním systémem android, určené pro automechanika, který bude vykonávat různé úkony. Pro testování aplikace jsem vymyslel úkon "dolití vody do nádržky od ostřikovačů".

V první kapitole jsem popsal jednotlivé základní pojimy, kdy bylo vytvořeno první rozhraní AR, jaké technické vybavení je potřeba pro vykreslování rozšířené reality, kde se rozšířená realita využívá a co to jsou markery.

V další části bakalářské práce jsem srovnal používané frameworky pro realizaci rozšířené reality. Ve třetí kapitole bylo seznámení s knihovnami OpenCV a Aruco a seznámení s Aruco markerama. Ve čtvrté kapitole byly popsány jednotlivé kroky, které se musí udělat před samostatnou detekcí markerů.

V páté kapitole jsem popisoval prostředí aplikace pro detekci markerů. Dále náplní této kapitoly byo popsání jednotlivých kroků úkonu, které by měl daný automechanik vykonat.

V šesté kapitole jsem testoval detekci markerů v různém prostředí ve kterém pracuej automechanik tzn. venkovní a vnitřní prostředí za dobrých i špatných světelných podmínek. Během testování jsem se setkal s hlavním problémem detekce markerů a to se špatnou detekcí markerů, důvody nedetekovaných markerů jsem popsal v poslední kapitole Zhodnocení testování 6.5.

Na základě jednotlivých výsledků testů jsem došel k závěru, že nejlepší umístění markerů pro detekci markerů je umístit markery mimo dosah světla/slunečního svitu. Také bylo potvrzeno tvrzení, že pro malé množství detekovaných markerů je dobré použít markery z menšího slovníku. V případě problému přímého světla na marker, kdy nastává přesvětlení markeru, by byla možnost implementace filtrů pro filtraci obrazu, tak aby filtr vyfiltroval přebytečné světlo z markeru.

Literatura

[1] *AUKSTAKALNIS*, Steve *Reálně O Virtuální Realitě: Umění a Věda Virtuální Reality.*

[2] Wikitude *Wikitude framework* [online]. Dostupný na WWW: `https://www.wikitude.com/`

[3] *OpenCV knihovna About.* [online] Dostupné na WWW: `https://opencv.org/`

[4] Vuforia *Vuforia framework* [online]. Dostupný na WWW: `https://www.vuforia.com/`

[5] *MALONEY*, Pete *Virtual Reality: Sword of Damocles.* [online]. [cit. 2017-03-26]. Dostupné na WWW: `https://cz.pinterest.com/pin/499407046152634584/`

[6] *AZUMA*, Ronald T. *A Survey of Augmented Reality.* (vlastní překlad autora).

[7] *Schroeder*, Stan *Google Glass Is Available to Everyone Today: How to Buy It.* [online]. [cit. 2014-04-15]. Dostupné na WWW: `https://mashable.com/2014/04/15/google-glass-how-to-buy/#BKfINWNPSZqt`

[8] *Raizian*, Ahmadreza *Head-mounted Displays (HMD).* [online]. [cit. 2016-03-14]. Dostupné na WWW: `http://ahmadrezarazian.ir/head-mounted-displays-hmd/`

[9] *Taş*, Yunus *What Are The Best Tools For Mobile AR App Development.* [online]. [cit. 2017-08-17]. Dostupné na WWW: `https://appsamurai.com/what-are-the-best-tools-for-mobile-ar-app-development/`

[10] *HORČÍK*, Jan *Překvapující technologie: rozšířená realita.* [online] 2009 [cit. 2012-03-31]. Dostupné na WWW: `http://pctuning.tyden.cz/multimedia/16-elektronika/15010-prekvapujici-technologie-rozsirena-realita?start=3`

[11] *Conn*, Mike *Niantic Is Breathing New Life Into 'Pokémon Go' With Its Newly Enhanced AR Feature.* [online] [cit. 2017-12-20]. Dostupné na WWW: `http://newscult.com/niantic-is-breathing-new-life-into-pokemon-go-with-its-newly-enhanced-ar-feature/`

[12] *HORČÍK*, Jan *Překvapující technologie: rozšířená realita.* [online] 2009 [cit. 31. 2012-03-31]. Dostupné na WWW: `http://pctuning.tyden.cz/multimedia/16-elektronika/15010-prekvapujici-technologie-rozsirena-realita?start=4`

[13] *Boeriu*, Horatiu *Head-Up Display 2.0 – Augmented Reality.* [online] [cit. 2011-10-7]. Dostupné na WWW: `http://www.bmwblog.com/2011/10/07/head-up-display-2-0-augmented-reality/`

[14] *Kudan*, Markerless Tracking *Markerless Tracking.* [online] [cit. 2017]. Dostupné na WWW: `https://www.kudan.eu/kudan-news/kudan-ar-tracking/`

[15] *Google ARCore Google ARCore.* [online] Dostupné na WWW: `https://developers.google.com/ar/discover/concepts`

[16] *Fiala*, Mark. *Designing Highly Reliable Fiducial Markers.* [online] [cit. 2009-07-31] Dostupné na WWW: `https://ieeexplore.ieee.org/document/5184844/`

[17] *Vumark VuMark.* [online] Dostupné na WWW: `https://library.vuforia.com/articles/Training/VuMark`

[18] *OpenCV knihovna About.* [online] Dostupné na WWW: `https://opencv.org/about.html`

[19] *Aruco knihovna Home.* [online] Dostupné na WWW: `https://www.uco.es/investiga/grupos/ava/node/1`

[20] *Marker 6x6 Marker 6x6* [online] Dostupné na WWW: `http://webnautes.tistory.com/1040`

[21] *Emami*,Shervin *Mastering OpenCV with Practical Vision Projects* [cit. 2012-11-27] 69 , ISBN - 10: 1849517827, ISBN - 13: 9781849517829

[22] *Atinfinity Detection of ArUco Markers* [online] [cit. 2017-10-10] Dostupné na WWW: `https://github.com/opencv/opencv_contrib/blob/master/modules/aruco/tutorials/aruco_detection/aruco_detection.markdown`

[23] *Sergarrido Detection of ArUco Markers* [online] [cit. 2015-10-26] Dostupné na WWW: `https://github.com/opencv/opencv_contrib/blob/master/modules/aruco/tutorials/aruco_detection/images/removeperspective.png`

[24] *Sergarrido Detection of ArUco Markers* [online] [cit. 2015-10-26] Dostupné na WWW: `https://github.com/opencv/opencv_contrib/blob/master/modules/aruco/tutorials/aruco_detection/images/bitsextraction1.png`

[25] *Sergarrido Calibration with ArUco and ChArUco* [online] [cit. 2015-10-26] Dostupné na WWW: `https://github.com/opencv/opencv_contrib/blob/master/modules/aruco/tutorials/aruco_calibration/images/arucocalibration.png`

[26] *Šonka*, Milan*Milan Šonka, Václav Hlaváč a Roger Boyle. Image Processing, Analysis and Machine Vision. 2* PWS, 1998. ISBN 0-534-95393-X

[27] *Emami*,Shervin *Mastering OpenCV with Practical Vision Projects* [cit. 2012-11-27], ISBN - 10: 1849517827, ISBN - 13: 9781849517829